Docket No. 1793.1075

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

Hyun-Kwon CHUNG, et al.

Serial No. 10/686,537 Group Art Unit: 2194

Confirmation No. 4036

Filed: October 16, 2003 Examiner: Nathan E. Price

For: DATA STORAGE MEDIUM HAVING INFORMATION FOR CONTROLLING BUFFERED STATE OF MARKUP DOCUMENT, AND METHOD AND APPARATUS FOR REPRODUCING DATA FROM THE DATA STORAGE MEDIUM

SUBMISSION OF ENGLISH TRANSLATION OF PRIORITY DOCUMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Pursuant to 37 CFR 1.55(a)(4) and MPEP 201.15, attached hereto are an English translation of Korean Patent Application No. 2002-63631 filed on October 17, 2002, one of the eight Korean priority applications of the present application, and a statement that the English translation is accurate to perfect the applicants' claim for foreign priority under 35 USC 119(a)-(d) with respect to this Korean priority application.

If there are any fees associated with filing of this paper, please charge the same to our Deposit Account No. 503333.

Respectfully submitted,

STEIN, MCEWEN & BUI, LLP

Date: 03/09/07

By: Randall S. Svihla

Randall S. Svihla
Registration No. 56,273

1400 Eye St., NW
Suite 300
Washington, D.C. 20005
Telephone: (202) 216-9505
Facsimile: (202) 216-9510

Attachments

CERTIFICATION OF TRANSLATION

I, ***Mi-sun Rhee***, an employee of Y.P. LEE, MOCK & PARTNERS of The Cheonghwa Bldg., 1571-18 Seocho-dong, Seocho-gu, Seoul, Republic of Korea, hereby declare under penalty of perjury that I understand the Korean language and the English language; that I am fully capable of translating from Korean to English and vice versa; and that, to the best of my knowledge and belief, the statement in the English language in the attached translation of ***Korean Patent Application No. 10-2002-0063631*** consisting of 41 pages, have the same meanings as the statements in the Korean language in the original document, a copy of which I have examined.

Signed this 19th day of October 2005

Rhee mi-sun

ABSTRACT

[Abstract of the Disclosure]

There are provided a data storage medium which contains information necessary for controlling the buffered state of a markup document, and a method and an apparatus for reproducing data from the data storage medium. The data storage medium includes: AV data which consists of audio data and/or video data; and buffering state information which indicates how much of a markup document has been buffered in order to prevent the AV data from being discontinuously displayed on a screen even when an apparatus for displaying the AV data including markup document data into which preload information used for issuing a command to read a file to be preloaded and store the read file in a memory is inserted cannot store the markup document information in a buffer due to physical damage to a disc or the malfunction of a network (e.g., discontinuity in the operation of the network). Accordingly, it is possible to reproduce AV data from a DVD using a markup document by outputting markup document data in conjunction with a moving image screen and taking advantage of substitute markup documents when the reproducing and downloading of data is delayed.

[Representative Drawing]

FIG. 14

SPECIFICATION

[Title of the Invention]

DATA STORAGE MEDIUM CONTAINING INFORMATION FOR CONTROLLING BUFFERED STATE OF MARKUP DOCUMENT, AND METHOD AND APPARATUS FOR REPRODUCING DATA FROM THE DATA STORAGE MEDIUM

[Brief Description of the Drawings]

The above and other features and advantages of the present invention will become more apparent by describing in detail exemplary embodiments thereof with reference to the attached drawings in which:

FIG. 1 is a diagram illustrating an interactive DVD on which AV data is recorded;

FIG. 2 is a diagram illustrating discontinuous reproduction of data from the interactive DVD shown in FIG. 1;

FIG. 3 is a block diagram of an apparatus for reproducing data from a data storage medium that carries out preloading or deleting markup documents according to a preferred embodiment of the present invention;

FIG. 4 is a diagram illustrating a directory structure of a DVD 300 that supports preloading or deleting markup documents according to the present invention;

FIG. 5 is a diagram illustrating a volume space of the DVD 300 that supports preloading or deleting markup documents according to the preferred embodiment of the present invention;

FIG. 6 is a flowchart for a process of preloading or deleting markup documents;

FIG. 7 is a flowchart for a process of interpreting preload information (step 602 of FIG. 6) according to a preferred embodiment of the present invention;

FIG. 8 is a flowchart for a process of preloading target files (step 603 of FIG. 6) according to a preferred embodiment of the present invention;

FIG. 9A is a flowchart for a process of preloading target files (step 603 of FIG. 6) according to another preferred embodiment of the present invention;

FIG. 9B is a flowchart for a process of preloading target files (step 603 of FIG. 6) according to still another preferred embodiment of the present invention;

FIG. 10 is a flowchart for a method of deleting at least one target file, which is to be preloaded and stored in memory, according to a preferred embodiment of the present invention;

FIG. 11 is a flowchart for a process of deleting a file, which is to be deleted from cache memory (step 1002 of FIG. 10), according to a preferred embodiment of the present invention;

FIG. 12 is a diagram illustrating the effects and advantages of a preloading technique according to a preferred embodiment of the present invention, in a case where AV data and HTML documents are recorded on an interactive DVD subjected to the preloading technique in the same manner shown in FIG. 1;

FIGS. 13 and 14 are block diagrams of an apparatus for reproducing data from a data storage medium according to a preferred embodiment of the present invention;

FIG. 15 is a diagram illustrating a process of managing a buffered state of a markup document in cache memory using a cache manager according to a preferred embodiment of the present invention;

FIG. 16 is a flowchart for a method of controlling the buffered state of a markup document using a content decoder and a cache manager, according to a preferred embodiment of the present invention;

FIG. 17 is a diagram illustrating an interactive DVD on which AV data and markup documents are recorded, according to a preferred embodiment of the present invention;

FIG. 18 is a diagram illustrating a directory structure of an interactive DVD on which AV data and markup documents are recorded in the same manner shown in FIG. 17;

FIG. 19 is a diagram illustrating the volume structure and file structure of an interactive DVD on which AV data and markup documents are recorded in the same manner shown in FIG. 17; and

FIG. 20 is a diagram illustrating a process of reproducing markup documents and AV data from the interactive DVD shown in FIG. 17, according to a preferred embodiment of the present invention.

[Detailed Description of the Invention]

[Object of the Invention]

[Technical Field of the Invention and Related Art prior to the Invention]

The present invention relates to a data storage medium containing information used for controlling the buffered state of a mark-up document, and a method and an apparatus for reproducing data from the data storage medium.

DVDs on which markup documents are recorded together with content (hereinafter, referred to as interactive DVDs) are now being commercialized in the market. Content recorded on an interactive DVD is reproduced in two different modes. One is a video mode, in which the content is displayed in the same manner as data recorded on a typical DVD, and the other is an interactive mode, in which the content is displayed through a display window defined by markup documents. When a user selects an interactive mode, a web browser installed in a DVD player displays the markup documents recorded on the interactive DVD. Content selected by the user is displayed through the display window defined by the mark-up documents.

For example, if the content is a movie title, a movie is displayed in the display window on a screen, and various pieces of additional information, such as the scenario, synopsis, and actors' and actresses' photos, can be displayed on the rest of the screen. Such additional information includes image files or text files.

FIG. 1 is a diagram illustrating an interactive DVD on which AV data is recorded. Referring to FIG. 1, AV data and a plurality of markup documents are recorded on tracks of the interactive DVD, in the form of an MPEG bitstream. Here, the markup

documents may include markup resources including various image files or graphic files for inserting into the markup documents.

FIG. 2 is a diagram illustrating discontinuous reproduction of data from the interactive DVD of FIG. 1. More specifically, FIG. 2 shows the occupancy of buffer memory, which is used for buffering AV data and occupancy of cache memory, which is used for caching web resources. Hereinafter, processes of loading AV data into memory and displaying the AV data will be described more fully with reference to FIGS. 1 and 2. A pickup device searches for a markup document STARTUP.HTM and loads the searched markup document STARTUP.HTM into the cache memory. Thereafter, STARTUP.HTM is activated. At the same time, AV data ① selected by a user is loaded into the buffer memory and then displayed. Thereafter, AV data ② is loaded into the buffer memory and then displayed. When buffering of the AV data ② is complete, the pickup device jumps to the place on the interactive DVD where AV data ③ is recorded and starts buffering the AV data ③. At this time, the user may request a markup document ④ A.HTM. Then, the pickup device stops buffering the AV data ③, searches for the markup document ④ A.HTM, and loads the markup document ④ A.HTM into the cache memory. While searching for the markup document ④ A.HTM and loading it into the cache memory, the AV data ③ is kept from being displayed. Therefore, the amount of data that can be buffered in the buffer memory drastically decreases because the AV data ③ still occupies space in the buffer memory. When the markup document ④ A.HTM is activated and the buffering of the AV data ③ is complete, AV data ⑤ is buffered. Thereafter, the pickup device jumps to the place where AV data ⑥ is recorded. At this point, all the data that has been buffered so far may disappear. In other words, in a case that requires the reproduction of DVD-video images from a conventional interactive DVD in synchronization with markup documents, for example, in a case that requires the display of a specific actor's or actress's personal history whenever he or she appears on a screen, the pickup device stops buffering AV data and begins searching for and caching the associated markup documents, and thus images may likely be discontinuously reproduced.

In the meantime, Korean Patent Application No. 02-57393 discloses a data storage medium storing preload information and an apparatus and method for reproducing data from the data storage medium. In this application, a preloading technique guaranteeing seamless reproduction of content in an interactive mode is disclosed. This preloading technique does not cause any problems when applied to preloading an entire group of files to be preloaded. However, according to this preloading technique, if some of the files to be preloaded are not properly read out, all of the files to be preloaded may not be able to be presented.

[Technical Goal of the Invention]

The present invention provides a data storage medium containing control information for controlling a buffered state of markup documents necessary for reproducing AV data so that the AV data can be presented in an appropriate manner in an interactive mode even though the markup documents have not been entirely preloaded, and an apparatus and a method for reproducing data from the data storage medium.

[Structure and Operation of the Invention]

According to an aspect of the present invention, there is provided a data reproduction apparatus which is capable of preloading a markup document necessary for reproducing AV data in an interactive mode. The data reproduction apparatus includes: a content decoder; and a cache manager. The content decoder generates a fetch signal, a retrieve signal, a release signal, a discard signal, and a report signal and outputs them to the cache manager.

The content decoder may generate the retrieve signal and displays a markup document on the screen of a display device connected to the data reproduction apparatus by interpreting and decoding the markup document.

The content decoder may generate the release signal after the markup document disappears from the screen of the display device.

The content decoder may generate the discard signal using a discard API when the markup document becomes of no use.

The content decoder may generate the report signal in order to determine a preloaded state of a markup document currently being preloaded using an IsCache API.

According to another aspect of the present invention, there is provided a data reproduction method in which AV data is reproduced in an interactive mode by preloading a markup document necessary for the reproduction of the AV data in the interactive mode. The data reproduction method includes preloading the markup document by generating a fetch signal, a retrieve signal, a release signal, a discard signal, and a report signal.

The data reproduction method may also include presenting the markup document by interpreting and decoding the markup document after the generation of the retrieve signal.

The release signal may be generated after the markup document disappears from the screen of the display device.

The discard signal may be generated using a discard API when the markup document becomes of no use.

The report signal may be generated in order to determine a preloaded state of a markup document currently being preloaded using an IsCache API.

The present invention will now be described in detail with reference to the accompanying drawings.

Preloading is a technique capable of preventing AV data from being reproduced with discontinuities due to the backlog of reading markup documents when reproducing the AV data together with the markup documents by loading the markup documents into a memory in advance. Detailed descriptions of the preloading technique and a technique of deleting preloaded are disclosed in Korean Patent Application No. 02-57393 filed on 19 September 2002.

FIG. 3 is a block diagram of an apparatus for reproducing data from a data storage medium that carries out preloading or deleting markup documents according to

a preferred embodiment of the present invention. Referring to FIG. 3, the apparatus for reproducing data from a data storage medium supports an interactive mode, in which an AV data stream is reproduced from a DVD 300 by decoding AV data recorded on the DVD 300 and then displaying the data in a display window defined by markup documents. The apparatus for reproducing data from a data storage medium includes a reader 1, first memory 2, second memory 3, an AV decoder 4, and a presentation engine 5. During an interactive mode, an AV screen is displayed while embedded in a markup screen. Markup documents are displayed in the markup screen, and the AV screen is obtained by reproducing AV data.

The presentation engine 5 supports extensions to link tags, JavaScript, or Java Applet, so that preload information written using link tags, the JavaScript application program interface (API), or the Java Applet API and deletion information written using the JavaScript API or the Java Applet API can be interpreted and executed, which will be described later in greater detail.

The reader 1 reads markup documents or AV data from the DVD 300. The first memory 2 is buffer memory, and buffers the AV data read by the reader 1. The second memory 3 is cache memory, and caches a received preload file. The AV decoder 4 decodes the AV data stored in the first memory 2 and outputs an AV data stream. The presentation engine 5 interprets the preload information, which is included in the markup documents read by the reader 1 and issues a request to the reader 1 or an Internet server (not shown) for files to be preloaded into the second memory 3 based upon the interpreted preload information. In the case requiring synchronized display of the files and the AV data, the preloaded files are read from the second memory 3 and displayed together with the AV data stream output from the AV decoder 4. Files are deleted from the second memory 3 by interpreting deletion information.

On the DVD 300, audio data or AV data is recorded, and the markup documents containing the preload information and/or the deletion information are also recorded.

In addition, a preload-list file and/or a deletion-list file may be further recorded on the DVD 300.

The preload-list file contains a list of files to be preloaded and the size of each file to be preloaded. The files to be preloaded represent markup documents, which are to be reproduced in synchronization with corresponding AV data. In the present embodiment, the files to be preloaded are recorded on the DVD 300. The files to be preloaded, however, may also be stored in an Internet server that is accessible through the Internet.

Preload information comprises a command to read the files to be preloaded from the DVD 300 and then store the files in cache memory 3. The preload information can be specified using a link tag, which contains the path and attributes of the preload-list file and is inserted into a head tag. Alternatively, the preload information can be specified using the JavaScript API or the Java Applet API, having the path and/or attribute of the preload-list file as function parameters and enabling the reproduction of the preload-list file. Alternatively, the preload information can be specified using the JavaScript API or the Java Applet API, having the path and/or attribute of each file to be preloaded as function parameters and enabling the reproduction of files, in which case the preload-list file is unnecessary.

The deletion-list file contains a list of files to be deleted, with the location information of each file to be deleted, i.e., the file name and path of each file to be deleted. The deletion information represents a command to delete files from the second memory 3. The deletion information can be specified using the JavaScript API or the Java Applet API having the location information of the deletion-list file as a function parameter and enabling the deletion of files that are listed on the deletion-list file. Alternatively, the deletion information can be specified using the JavaScript API or the Java Applet API having the location information of each file to be deleted as a function parameter and enabling the deletion of files, in which case the deletion-list file is unnecessary.

FIG. 4 is a diagram illustrating the directory structure of the DVD 300 according to the present invention. Referring to FIG. 4, the root directory has a DVD video directory VIDEO_TS containing AV data and a DVD interactive directory DVD_ENAV containing data for supporting an interactive function.

Header information VIDEO_TS.IFO concerning all video titles recorded on the DVD 300, navigation information VTS_01_0.IFO for a first video title, and AV data VTS_01_0.VOB, VTS_01_1.VOB, ... constituting the first video title are recorded in the DVD video directory VIDEO_TS. The detailed description of the structure of the DVD video directory VIDEO_TS is disclosed in the DVD-Video standard, "DVD-Video for Read Only Memory Disc 1.0".

Navigation information DVD_ENAV.IFO regarding the entire interactive information and a start-up document STARTUP.HTM are recorded in the DVD interactive directory DVD_ENAV. In addition, a preload-list file STARTUP.PLD, a file to be preloaded A.HTM, and a graphic file A.PNG inserted into A.HTM are also provided in the DVD interactive directory DVD_ENAV. Other files to be preloaded or graphic files inserted thereinto may also be recorded in the DVD interactive directory DVD_ENAV.

FIG. 5 is a diagram illustrating the volume space of the DVD 300 according to the present invention. Referring to FIG. 5, the volume space includes a control information region which contains control information for the volume space of the DVD 300 and files recorded on the DVD 300, a DVD-Video data region where video title data is recorded, and a DVD-Interactive data region which is provided for reproducing AV data during interactive mode.

The files stored in the DVD video directory VIDEO_TS of FIG. 4, i.e., VIDEO_TS.IFO, VTS_01_0.IFO, VTS_01_0.VOB, VTS_01_1.VOB, ..., are recorded in the DVD-Video data region. The files stored in the DVD interactive directory DVD_ENAV, i.e., STARTUP.HTM, STARTUP.PLD, A.HTM, and A.PNG, are recorded in the DVD-Interactive data region.

A method for reproducing data from a data storage medium according to a preferred embodiment of the present invention will be described in the following paragraphs in greater detail.

FIG. 6 is a flowchart for a method of reproducing data from a data storage medium according to a preferred embodiment of the present invention. Referring to FIG. 6, when the interactive mode is selected, the reader 1 reads an HTML document, which is a markup document recorded on the DVD 300, from the DVD 300 in step 601. The presentation engine 5 interprets preload information contained in the HTML document and requests that the reader 1 or an Internet server preload files in step 602. In response to the request, files to be preloaded are stored in the second memory 3 in step 603.

The reader 1 reads AV data, corresponding to the HTML document read in step 601, from the DVD 300 and stores the read AV data in the first memory 2, which is buffer memory, in step 604. The AV decoder 4 decodes AV data stored in the first memory 2 into an AV data stream in step 605. In step 606, the presentation engine 5 reads the preloaded files from the second memory 3 and displays the decoded AV data stream in a display window, which is defined by the HTML document read by the reader 1 in step 601.

FIG. 7 is a flowchart for an example of step 602 of FIG. 6, in which preload information is interpreted. Referring to FIG. 7, the presentation engine 5 recognizes the path of a preload-list file included in an HTML document in step 701 and reads the preload-list file by following the recognized path in step 702. Thereafter, the presentation engine 5 recognizes the files to be preloaded, which are listed in the preload-list file, in step 703. Here, recognition of the files to be preloaded indicates recognition of the paths and attributes of the files to be preloaded.

FIG. 8 is a flowchart for an example of step 603 of FIG. 6, in which files are preloaded. Referring to FIG. 8, the presentation engine 8 interprets the preload-list file, containing a preload tag that has the paths and attributes of the files to be preloaded as parameters, and performs preloading of the files in step 802.

FIG. 9A is a flowchart for another example of step 603 of FIG. 6, in which files are preloaded. Referring to FIG. 9A, the presentation engine 5 interprets the API inserted into a body tag using parameters specifying the paths of the files to be preloaded and reads the files to be preloaded using the API in step 901a. Since the presentation engine 5 can determine the attributes of the files to be preloaded, it can process the files to be preloaded based on their attributes and store the processed files in memory.

FIG. 9B is a flowchart for still another example of step 603 of FIG. 6. Referring to FIG. 9B, in step 901b, the presentation engine 5 calls an API inserted into a body tag having the path and attribute of a file to be preloaded as parameters, and stores the file to be preloaded in memory. Since the attribute of the file to be preloaded can be identified, the presentation engine 5 can process the file to be preloaded in consideration of its attribute and then can store the file to be preloaded in memory.

FIG. 10 is a flowchart for a method of deleting at least one of the preloaded files that are stored in memory, according to a preferred embodiment of the present invention. Referring to FIG. 10, the presentation engine 5 interprets deletion information contained in an HTML document in step 1001, identifies files to be deleted based on a deletion-list file, and deletes the identified files from the second memory 3 in step 1002. In the present embodiment, the preload-list file and the deletion-list file are integrated into a single file, i.e., STARTUP.PLD, which will be described later in greater detail. However, it is obvious to one skilled in the art that a list of files to be preloaded and a list of files to be deleted can be realized as two separate files rather than being integrated into a single file.

FIG. 11 is a flowchart for an example of step 1002 of FIG. 10, in which files are deleted from cache memory. Referring to FIG. 11, a list of files to be deleted are recorded in the deletion-list file and the files are deleted from the second memory 3 using an API, having the path of the deletion-list file as a parameter, in step 1101. Here, the deletion of the files may be a process of physically removing files from the second memory 3, or a process of including in the files a flag indicating that the files can

be deleted from the second memory 3 or can be overwritten by other data without physically removing the files from the second memory 2.

FIG. 12 is a diagram illustrating the effects of a preloading process on an Interactive DVD where AV data and HTML documents are recorded in the same manner as in FIG. 1, according to a preferred embodiment of the present invention. FIG. 12 shows occupancy of the first memory 2 where MPEG-coded AV data is buffered and occupancy of the second memory 3 where a web resource is cached. Referring to FIGS. 1 and 12, the reader 1 searches for and reads STARTUP.HTM, and the presentation engine 5 interprets the preload information included in STARTUP.HTM so that ④ A.HTM is preloaded into the second memory 3. When STARTUP.HTM, which is loaded into the second memory 3, is activated, ① AV data is loaded into the first memory 2 and then displayed. Thereafter, ② AV data is loaded into the first memory 2 and begins then displayed. When buffering of ② AV data is complete, the reader 1 jumps to the place where ③ AV data is recorded and starts buffering ③ AV data. At this time, if a user requests ④ A.HTM, the presentation engine 5 reads ④ A.HTM from the second memory 3 and displays ④ A.HTM. In this case, there is no need for the reader 1 to stop the buffering of ③ AV data, search the DVD 300 for ④ A.HTM, and then load the document ④ A.HTM into the second memory 3. Therefore, the reader 1 can continue buffering ③ AV data. If the reader 1 completes buffering of ⑤ AV data and then jumps to the place where ⑥ AV data is recorded, the amount of data buffered in the first memory 2 may be reduced. However, the amount of data that has been buffered in the first memory 2 is sufficient so that a shortage in buffered data does not occur. In other words, even when there is a need to display DVD-video images, reproduced from an interactive DVD during interactive mode, in synchronization with HTML documents, the reader 1 does not have to stop the buffering of AV data and then search for and cache the HTML documents, because the HTML documents have already been preloaded in the second memory 3. Synchronization display could be used when there is a need to display a specific actor's or actress's personal history whenever he or she appears on a screen.

The above-mentioned processes of preloading data and deleting the preloaded data are taught by the applicant of this disclosure in Korean Patent Application No. 02-57393 entitled "Data Storage Medium Containing Preload Information, and Apparatus and Method for Reproducing Data from the Data Storage Medium" (filed on September 19, 2002).

Hereinafter, a data storage medium and a method and apparatus for reproducing data from the data storage medium, which are capable of appropriately controlling a buffered state of markup documents so that AV data can be appropriately presented even though some of the markup documents have not yet been read, will be described in greater detail.

FIG. 13 is a block diagram of an apparatus for reproducing data from a data storage medium according to a preferred embodiment of the present invention. Referring to FIG. 13, the apparatus for reproducing data from a data storage medium, like the apparatus illustrated in FIG. 3. In addition, the apparatus for reproducing data from a data storage medium of FIG. 13 supports an interactive mode, carries out preloading, and includes an AV buffer 20, an AV reproduction engine 40, an ENAV buffer 30, and an ENAV engine 50.

The AV buffer 20, which corresponds to first memory 2 of FIG. 3, buffers AV data read from a disk 100 or a network. The AV reproduction engine 40 decodes the buffered AV data, thus outputting an AV stream. The ENAV buffer 30, which is cache memory corresponding to second memory 3 of FIG. 3, buffers markup documents read from the disk 100 or the network. The ENAV engine 50, which corresponds to the presentation engine 5 of FIG. 3, carries out preloading and controls a buffered state of markup documents stored in the ENAV buffer 30. In addition, the ENAV engine 50 interprets or decodes the markup documents stored in the ENAV buffer 30 so that the AV stream output from the AV reproduction engine 40 can be reproduced in an interactive mode.

FIG. 14 is a detailed block diagram of the ENAV engine 50 of FIG. 13. Referring to FIG. 14, the ENAV engine 50 includes a buffer manager 51 that controls the ENAV buffer 30 and a content decoder 52 that interprets markup documents.

The content decoder 52 includes an interpretation engine that parses and interprets markup documents and a browser that draws markup documents from the interpretation engine and the network. Here, the markup documents correspond to various kinds of markup resources, ranging from markup text data written in HTML, CSS, or JAVASCRIPT to binary data, such as image data, audio data, or a Java program, which is referred to by markup documents. The markup documents are drawn from the disk 100 or the network by the buffer manager 51 in the ENAV engine 50.

In the case of preloading or deleting markup documents, the buffer manager 51 manages a buffered state of the markup documents in a predetermined manner according to the present invention. More specifically, the buffer manager 51 operates in different ways in response to five different signals input from the content decoder 52.

FIG. 15 is a diagram illustrating the buffer manager 51 managing a buffered state of markup documents processed by the ENAV buffer 30. Referring to FIG. 15, five different signals, i.e., a fetch signal, a retrieve signal, a release signal, a discard signal, and a report signal, are input into the buffer manager 51 from the content decoder 52.

A fetch signal is used to preload markup documents into the ENAV buffer 30. If the markup documents are already preloaded into the ENAV buffer 30, an I/O manager prevents the corresponding markup documents from being read from a disk or a network. The I/O manager represents a reader (not shown), which reads data from the disk, or a network data receiver/transmitter (not shown), which receives data from the network. The reader reads files from the disk, and the network data receiver/transmitter receives predetermined data from or transmits predetermined data to the network using the HTTP protocol.

Referring to FIG. 15, the I/O manager is set to operate in the following manners. When an HTTP request is issued, the I/O manager uses unblocked I/O. On the other

hand, when a request for files on a disk is issued, the I/O manager uses blocked I/O. In the case of reproducing markup documents from a network, the I/O manager adopts an unblocked method so that a plurality of markup documents can be received at a given time. However, if a plurality of markup documents is read from a disk at a given time, a pickup device (not shown) in the reader is required to move between locations where the plurality of markup documents are recorded, so many times that the speed of reading the corresponding markup documents is lowered. Therefore, in the case where a plurality of markup documents needs to be read from a disk, a sequential blocked I/O process is adopted, in which the plurality of markup documents are sequentially read from the disk.

A retrieve signal is used when issuing a request for transferring data from the ENAV buffer 30 to the content decoder 52. Therefore, when predetermined data is read from a disk or downloaded from a network, the content decoder 52 is blocked from operating until reading or downloading of the predetermined data is completed.

A release signal indicates that the predetermined data transferred from the ENAV buffer 30 to the content decoder 52, in response to the above-described retrieve signal, is no longer needed. Therefore, if a predetermined markup document is referred to five times in response to a retrieve signal, a release signal is generated five times. A counter value increases whenever a retrieve signal is generated and decreases whenever a release signal is generated. When a counter value corresponding to a predetermine markup document reaches 0, i.e., when all reproduced markup documents are released, the released markup documents are deleted from the ENAV buffer 30 in response to a discard signal, which will be described in greater detail in the following paragraph.

A discard signal indicates that predetermined markup documents stored in the ENAV buffer 30 can be deleted from the ENAV buffer 30 because they will not be used any more. Therefore, in response to the discard signal, the predetermined markup documents are discarded from the ENAV buffer 30. If markup documents are associated with another preferred application and the retrieve signal has been

generated, but the release signal has not yet been generated, the markup documents cannot be deleted from the ENAV buffer 30, even when a discard signal has been generated by a predetermined application.

A report signal is used to verify whether or not markup documents read in response to a fetch signal are successfully loaded into the ENAV buffer 30, whether or not at least some of the corresponding markup documents cannot be read due to errors, or whether the corresponding markup documents are being read.

The following is a description of APIs used in scripts that are written in a markup document.

1. [obj].preload(URL, resType)

1) Description

This API is used for preloading files, or reading files and storing the files in the ENAV buffer 30. Parameters of the corresponding API include location information for the preload-list file or files to be preloaded, and location information and attributes of the files to be preloaded. The corresponding API generates a fetch signal and can be applied to all files that can be read from a disk (disc://) or a network (http://).

2) Parameters

URL = : path of the preload-list file or paths of the files to be preloaded

ResType = : attributes of the files to be preloaded

3) Return values

If the files to be preloaded are successfully preloaded into the ENAV buffer 30, a value of 0 is returned. Otherwise, a value of –1 is returned.

4) Examples

navigator.preload("disc://dvd_enav/a.htm", "text/xml"), which indicates a request for loading files from "disc://dvd_enav/a.htm". The files to be preloaded are text files written in XML.

navigator.preload("disc://dvd_enav/a.pld", "xml/preload"), which represents a request for loading files listed in a preload-list file from "disc://dvd_enav/a.pld". The files listed in the preload-list file are preload files written in XML.

2. [obj].discard(URL, resType)

1) Description

This API is used for deleting files from the ENAV buffer 30. Parameters of the corresponding API include location information for the deletion-list file or files to be deleted and attributes of the files to be deleted. The corresponding API generates a discard signal.

2) Parameters

URL = : path of the deletion-list file or paths of the files to be deleted

ResType = : attributes of the files to be deleted

3) Return values

If a preload command is successfully issued, a value of 0 is returned. Otherwise, a value of –1 is returned.

4) Examples

navigator.discard("disc://dvd_enav/a.htm", "text/xml"), which represents a request for deleting files, indicated by a deletion-list file of "disc://dvd_enav/a.pld", from the cache memory. The corresponding files are list files written in XML.

3. [obj].isCached(URL, resType)

1) Description

This API is used for checking whether files to be searched for have been successfully stored in the ENAV buffer 30. Parameters of the corresponding API include location information for the search list file or files to be searched for, and attributes of the files to be searched for. The corresponding API generates a report signal and can be applied to all files that are read from a disk (disc://) or a network (http://).

2) Parameters

URL = : path of the search list file or paths of the files to be searched for

resType = : attributes of the files to be searched for

3) Return values

If files listed in the search list file or the files to be searched for are successfully stored in the ENAV buffer 30, a value of 0 is returned. Otherwise, a value of –1 is returned. If all the files to be searched for other than the one(s) currently being read by the ENAV buffer 30, have already been successfully stored in the ENAV buffer 30, a value of –2 is returned.

4) Examples

navigator.isCached("disc://dvd_enav/a.htm", "text/xml"), which represents a request for verifying whether or not a file of "disc://dvd_enav/a.htm" has already been stored. The corresponding file is a text file written in XML.

navigator.isCached("disc://dvd_enav/a.pld", "xml/preload", which represents a request for verifying whether or not files, referred to by the list file of "disc://dvd_enav/a.pld", have already been stored. The corresponding files are list files written in XML.

4. [obj].progressNameOfFile

1) Description

This API is a value that returns a universal resource identifier (URI) of a file currently being preloaded to a predetermined value.

2) Return value: file path or a URI

5. [obj].progressLengthOfFile

1) Description

This API indicates how much of the file currently being preloaded has been preloaded.

2) Return value: a value represented in a byte unit

6. [obj].remainLengthOfFile

1) Description

This API indicates how much of the file currently being preloaded is yet to be preloaded.

2) Return value: a value represented in a byte unit

7. [obj].totalLoadingSize

1) Description:

This API indicates a total load of files to be preloaded.

2) Return value: a value represented in a byte unit

8. [obj].remainLoadingSize

1) Description

This API indicates how much of the total load of files to be preloaded is yet to be dealt with.

2) Return value: a value represented in a byte unit

9. [obj].allDone

1) Description

This API indicates whether or not an apparatus for reproducing data from a data storage medium has completed preloading.

2) Return values

If the apparatus for reproducing data from a data storage medium has successfully completed preloading, this API returns to 'True'. Otherwise, this API returns to 'False'.

As described above, the retrieve signal and the release signal are generated whenever corresponding markup documents are used. For example, the content

decoder 52 presents an image of "disc://dvd_enav/a.png" using a display device (not shown) by interpreting and generating a retrieve signal so that the buffer manager 51 reproduces the corresponding image from the ENAV buffer 30. Likewise, the content decoder 52 generates a release signal when the presentation of the corresponding image is complete.

FIG. 16 is a flowchart for a method of controlling the buffer state carried out by the content decoder 52 and the buffer manager 51, according to a preferred embodiment of the present invention.

Referring to FIG. 16, in step 1601, a content decoder receives a preload command and then generates a fetch signal. In step 1602, a buffer manager receives the fetch signal and starts to read predetermined markup documents. In step 1603, the content decoder checks whether or not error occurs during the reading of the markup documents. In step 1604, if error occurs, the content decoder processes error. During checking whether error occurs, the content decoder transmits a report signal to the buffer manager. In step 1605, the buffer manager transmits a signal to the content decoder in response to the reception of the report signal. In step 1606, the content decoder generates a retrieve signal after reading up the markup documents and then transmits the retrieve signal to the buffer manager in order to use the markup documents. In step 1607, the buffer manager receives the retrieve signal from the content decoder and transmits designated markup documents to the content decoder. In step 1608, the content decoder presents the designated markup documents received from the buffer manager. After the presentation of the designated markup documents, the content decoder generates a release signal and transmits the release signal to the buffer manager in step 1609 because once presented, the designated markup documents are not be used any longer. In step 1610, the buffer manager receives the release signal from the content decoder and decreases a current value of a counter that indicates how many times the markup documents have been used by 1. In step 1611, the content decoder generates a discard signal and transmits the discard signal to the

buffer manager. In step 1612, the buffer manager deletes the designated markup documents from an ENAV buffer in response to the reception of the discard signal.

FIG. 17 is a schematic diagram illustrating a disk on which AV data and markup documents are recorded, according to a preferred embodiment of the present invention. Referring to FIG. 17, AV data and markup documents are recorded on a disk along tracks. A startup document STARTUP.HTM contains a preload list file STARTUP.PLD. STARTUP.PLD is a preload list file provided for seamlessly reproducing files ranging from A.HTM to D.HTM.

FIG. 18 is a diagram illustrating the directory structure of the disk of FIG. 17. Referring to FIG. 18, reference documents of the startup document STARTUP.PLD are included in a directory DVD_ENAV.

FIG. 19 is a diagram illustrating the volume structure and file structure of the disk of FIG. 17. Referring to FIG. 19, all the reference documents of STARTUP.PLD are recorded in a DVD interactive data area.

FIG. 20 is a diagram illustrating a predetermined order in which the markup documents and the AV data recorded on the disk of FIG. 17 are reproduced. Referring to FIG. 20, when each scene begins, it is checked using IsCashed( ) whether or not files that a preload list file for each scene refers to have already been read. If reading of all the reference files of the preload list file has been successfully completed, HTM documents are read and reproduced. Thereafter, markup documents that have already been reproduced are discarded using Discard( ).

FIG. 18 is a diagram illustrating the directory structure of the disk of FIG. 17, FIG. 19 is a diagram illustrating the volume structure and file structure of the disk of FIG. 17, and.

In order to seamlessly reproduce data from STARTUP.HTM, A.HTM, and D.HTM, STARTUP.PLD must be specified as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PRELOAD PUBLIC "-//DVD//DTD DVD Preload List 1.0//EN"
"http://www.dvdforum.org/enav/dvd-preload-list.dtd"-->
<filedef type="text/xml" src="disc://dvd_enav//a.htm" />
<filedef type="text/xml" src="disc://dvd_enav//a.pld" />
<filedef type="image/png" src="dvd://dvd_enav//a1.png" />
<filedef type="image/png" src="dvd://dvd_enav//a2.png" />
<filedef type="image/png" src="dvd://dvd_enav//a3.png" />
<filedef type="text/xml" src="disc://dvd_enav//b.htm" />
<filedef type="text/xml" src="disc://dvd_enav//b.pld" />
<filedef type="audio/au" src="dvd://dvd_enav//b1.au" />
<filedef type="image/png" src="dvd://dvd_enav//b2.png" />
<filedef type="image/png" src="dvd://dvd_enav//b3.jpg" />
<filedef type="text/xml" src="disc://dvd_enav//c.htm" />
<filedef type="text/xml" src="disc://dvd_enav//c.pld" />
<filedef type="image/png" src="dvd://dvd_enav//c1.png" />
<filedef type="image/png" src="dvd://dvd_enav//c2.png" />
<filedef type="image/png" src="dvd://dvd_enav//c3.png" />
<filedef type="text/xml" src="disc://dvd_enav//d.htm" />
<filedef type="text/xml" src="disc://dvd_enav//d.pld" />
<filedef type="image/png" src="dvd://dvd_enav//d1.png" />
<filedef type="image/png" src="dvd://dvd_enav//d2.png" />
</preload>
```

By using STARTUP.PLD above, STARTUP.HTM is displayed on a screen, indicating the start of the interactive presentation. An example of STARTUP.HTM, which is processed by the apparatus for reproducing data from a data storage medium of FIG. 15, is as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC -//DVD/DTD XHTML DVD-HTML1.0//EN"
"http://www.dvdforum.org/enav/dvdhtml-1-0.dtd">
<html>
<head>
<title>WAR II STARTUP PAGE</title>
<script language="ecmascript">
<![CDATA[
function onload_handler ()
{
navigator.preload("disc://dvd_enav/startup.pld","xml/preload");
idplayer.subscribeToEvent(10)
idplayer.setTrigger(1,"00:30:35:00",1);
idplayer.play();
docbody.addEventListener("dvdevent",idplayer_handler,true);
}
function idplayer_handler(e)
{
switch(e.parm1)
{
case 10: // trigger event
   if (e.parm2 == 1) // begin to die
   {
      while (navigator.isCached("disc://dvd_enav/a.pld","xml/preload") == 2
|| navigator.isCached("disc://dvd_enav/b.pld","xml/preload") == 2
|| navigator.isCached("disc://dvd_enav/c.pld","xml/preload") == 2
|| navigator.isCached("disc://dvd_enav/d.pld","xml/preload") == 2); // during
//reading;
       if (navigator.isCached("disc://dvd_enav/a.pld","xml/preload") == 1) // failed
       {
         idplayer.stop();
         location.href = "disc://dvd_enav/discerr.htm";
       }
      // to read c.pld is OK.
location.href = "disc://dvd_enav/a.htm"; // jump to c.htm
}
break;
   }
}
]]>
</script>
</head>
<body id="docbody" onload="onload_handler ()">
<object style="position: absolute; left: 150px; top: 100px; width: 370px; height: 250px"
data="dvd:video_ts" id="idplayer"/>
<img style="position: absolute; left: 167px; top: 375px; width: 170px; height: 60px" src="
disc://dvd_enav/s1.png" type="image/png"/>
<img style="position: absolute; left: 370px; top: 375px; width: 170px; height: 60px" src="
disc://dvd_enav/s2.png" type="image/png"/>
</body>
</html>
```

The markup documents A.HTM and B.HTM include images. Here, all markup documents necessary for presenting A.HTM, i.e., all markup documents in A.PLD and mentioned as files to be preloaded, are deleted from the ENAV buffer 30.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC -//DVD/DTD XHTML DVD-HTML1.0//EN"
"http://www.dvdforum.org/enav/dvdhtml-1-0.dtd">
<html>
<head>
<title>WAR II B.HTM PAGE</title>
<script language="ecmascript">
<![CDATA[
function onload_handler ()
{
navigator.discard ("disc://dvd_enav/a.pld","xml/preload"); // any longer to use A.HTM
idplayer.subscribeToEvent(10)
idplayer.setTrigger(1,"50:35:00",1);
docbody.addEventListener("dvdevent",idplayer_handler,true);
}
function idplayer_handler(e)
{
switch(e.parm1)
{
case 10: .// trigger event
   if (e.parm2 == 1) // begin combat
   {
      while (navigator.isCached("disc://dvd_enav/c.pld","xml/preload") == 2); // during
        //reading;
       if (navigator.isCached("disc://dvd_enav/c.pld","xml/preload") == 1) // failed
       {
         idplayer.stop();
         location.href = "disc://dvd_enav/discerr.htm";
       }
      // to read a.pld is OK.
location.href = "disc://dvd_enav/c.htm"; // jump to c.htm
}
break;
   }
}
]]>
</script>
</head>
<body id="docbody" onload="onload_handler ()">
<object style="left: 110px; top: 80px; width: 500px; height: 200px" data="dvd:video_ts"
id="idplayer"/>
<img style="position: absolute; left: 539px; top: 38px; width: 140px; height: 70px" src="
disc://dvd_enav/b1.png" type="image/png" />
<img style="position: absolute; left: 560px; top: 200px; width: 120px; height: 50px" src="
disc://dvd_enav/b2.png" type="image/png" />
<img style="position: absolute; left: 610px; top: 280px; width: 100px; height: 50px" src="
disc://dvd_enav/b3.png" type="image/png" />
</body>
</html>
```

[Effect of the Invention]

According to the present invention, images can be presented using only preloaded files because content is processed using a predetermined method that enables determination of the preloaded content state, even when physical defects of a

disk or connection disruptions cause unsuccessful or incomplete preloading of files into a buffer. Therefore, it is possible to enhance the reliability of reproducing content in the presence of errors that may occur during reproduction of the content.

What is claimed is:

1. A data reproduction apparatus which is capable of preloading a markup document necessary for reproducing AV data in an interactive mode, the data reproduction apparatus comprising:

a content decoder; and

a cache manager,

wherein the content decoder generates a fetch signal, a retrieve signal, a release signal, a discard signal, and a report signal and outputs them to the cache manager.

2. The data reproduction apparatus of claim 1, wherein the content decoder generates the retrieve signal and displays a markup document on the screen of a display device connected to the data reproduction apparatus by interpreting and decoding the markup document.

3. The data reproduction apparatus of claim 1, wherein the content decoder generates the release signal after the markup document disappears from the screen of the display device.

4. The data reproduction apparatus of claim 1, wherein the content decoder generates the discard signal using a discard API when the markup document becomes of no use.

5. The data reproduction apparatus of claim 1, wherein the content decoder generates the report signal in order to determine a preloaded state of a markup document currently being preloaded using an IsCache API.

6. A data reproduction method in which AV data is reproduced in an interactive mode by preloading a markup document necessary for the reproduction of the AV data in the interactive mode, the data reproduction method comprising

preloading the markup document by generating a fetch signal, a retrieve signal, a release signal, a discard signal, and a report signal.

7. The data reproduction method of claim 6 further comprising presenting the markup document by interpreting and decoding the markup document after the generation of the retrieve signal.

8. The data reproduction method of claim 6, wherein the release signal is generated after the markup document disappears from the screen of the display device.

9. The data reproduction method of claim 6, wherein the discard signal is generated using a discard API when the markup document becomes of no use.

10. The data reproduction method of claim 6, wherein the report signal is generated in order to determine a preloaded state of a markup document currently being preloaded using an IsCache API.

FIG. 1

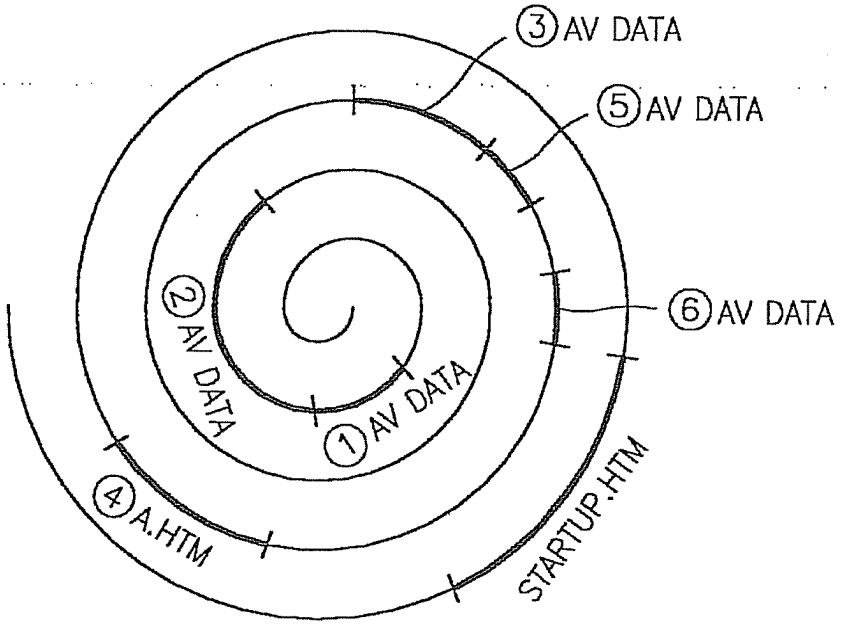

③AV DATA
⑤AV DATA
⑥AV DATA
②AV DATA
①AV DATA
④A.HTM
STARTUP.HTM

FIG. 2

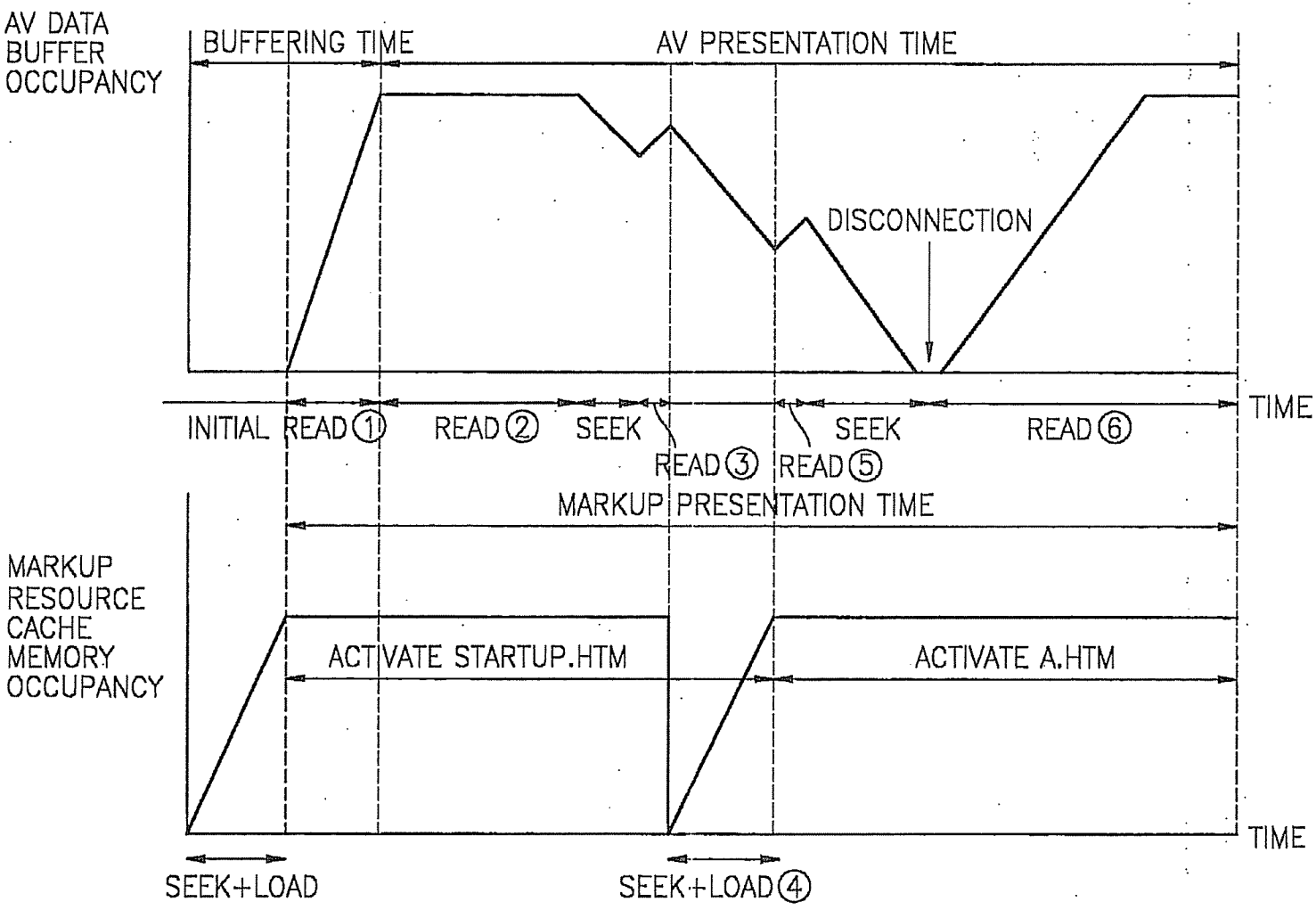

AV DATA BUFFER OCCUPANCY
BUFFERING TIME
AV PRESENTATION TIME
DISCONNECTION
TIME
INITIAL READ ①
READ ②
SEEK
READ ③
READ ⑤
SEEK
READ ⑥
MARKUP PRESENTATION TIME
MARKUP RESOURCE CACHE MEMORY OCCUPANCY
ACTIVATE STARTUP.HTM
ACTIVATE A.HTM
TIME
SEEK+LOAD
SEEK+LOAD ④

FIG. 3

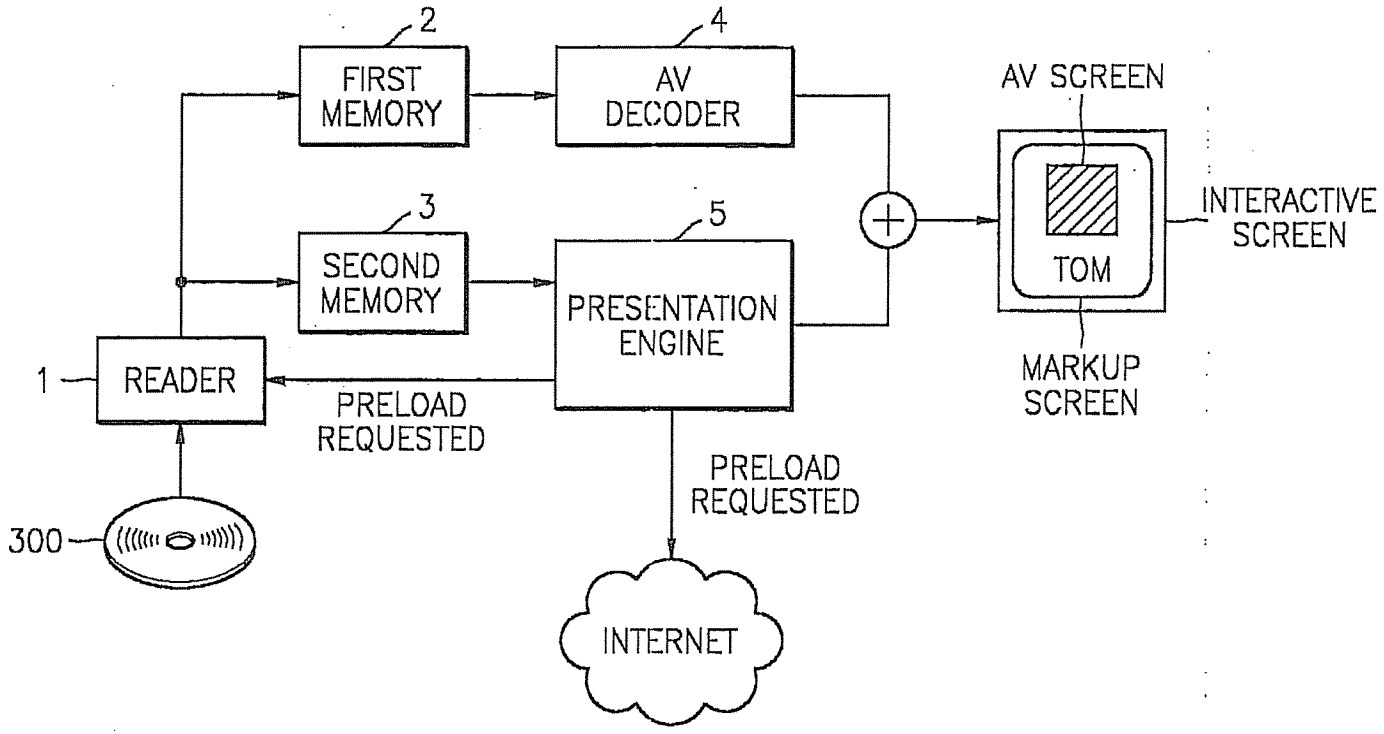

2
FIRST MEMORY
4
AV DECODER
AV SCREEN
3
SECOND MEMORY
5
PRESENTATION ENGINE
TOM
INTERACTIVE SCREEN
1
READER
PRELOAD REQUESTED
MARKUP SCREEN
PRELOAD REQUESTED
300
INTERNET

FIG. 4

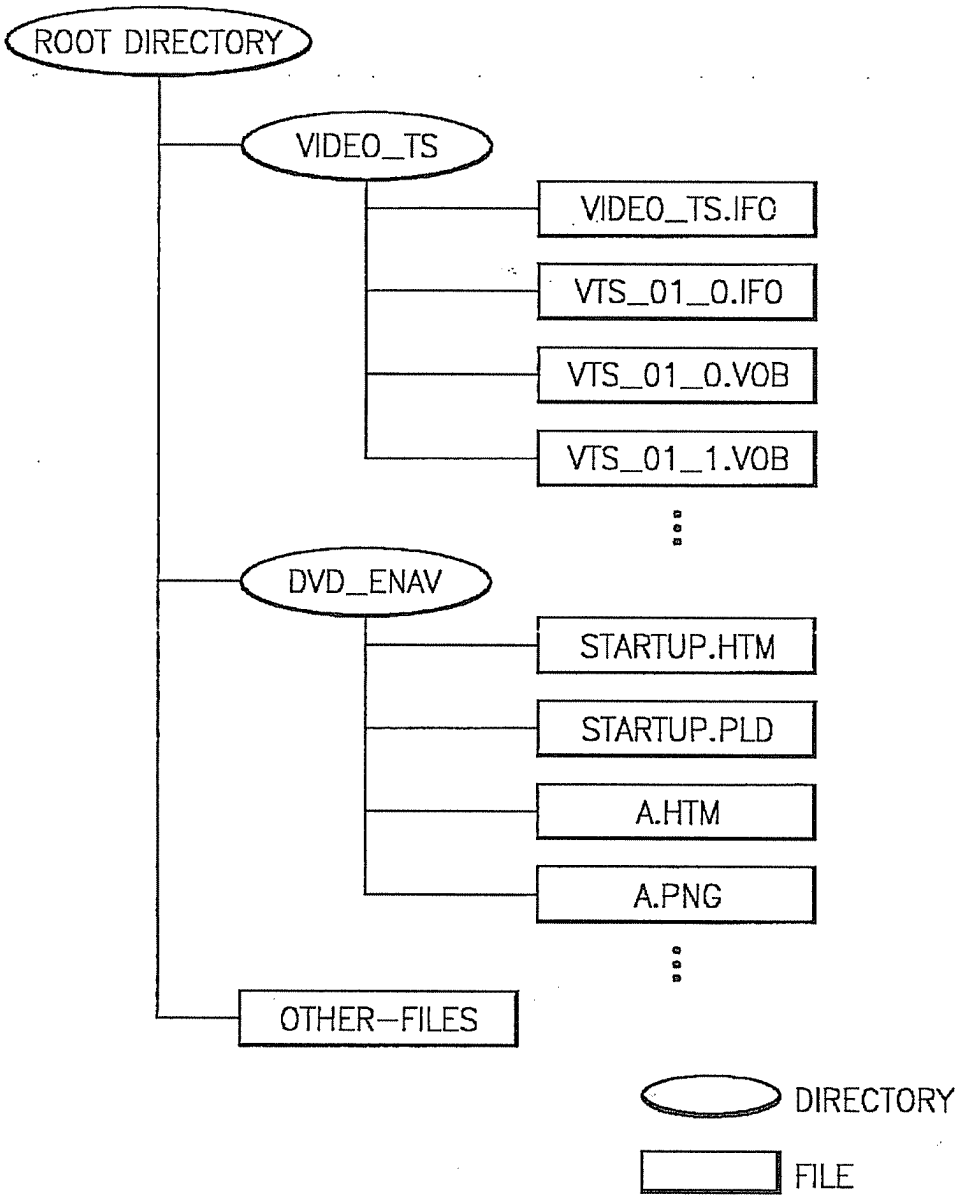

ROOT DIRECTORY
VIDEO_TS
VIDEO_TS.IFO
VTS_01_0.IFO
VTS_01_0.VOB
VTS_01_1.VOB
DVD_ENAV
STARTUP.HTM
STARTUP.PLD
A.HTM
A.PNG
OTHER-FILES
DIRECTORY
FILE

FIG. 5

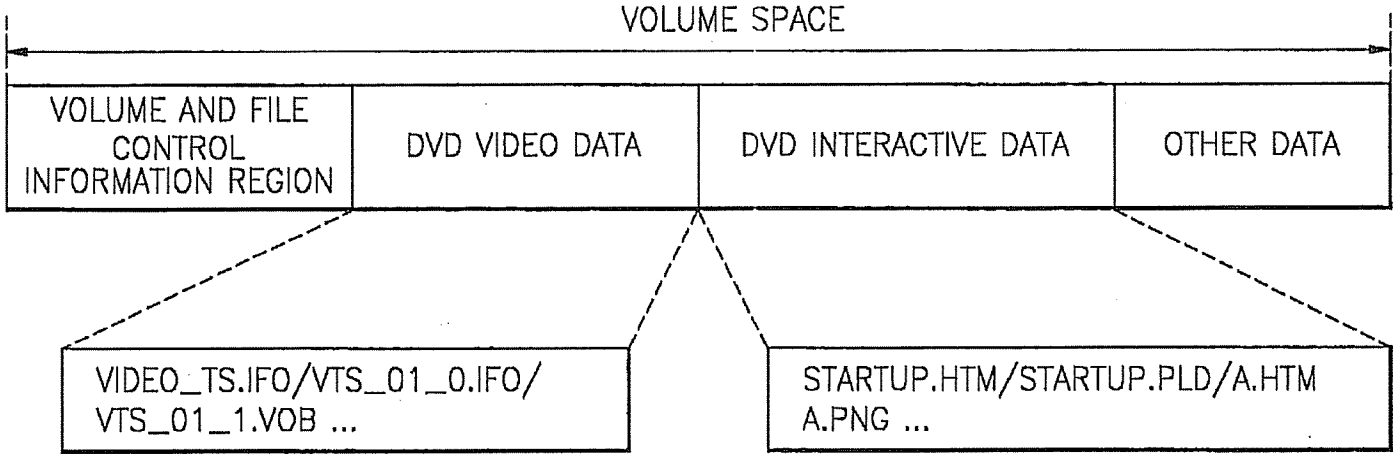

VOLUME SPACE
VOLUME AND FILE CONTROL INFORMATION REGION
DVD VIDEO DATA
DVD INTERACTIVE DATA
OTHER DATA
VIDEO_TS.IFO/VTS_01_0.IFO/
VTS_01_1.VOB ...
STARTUP.HTM/STARTUP.PLD/A.HTM
A.PNG ...

FIG. 6

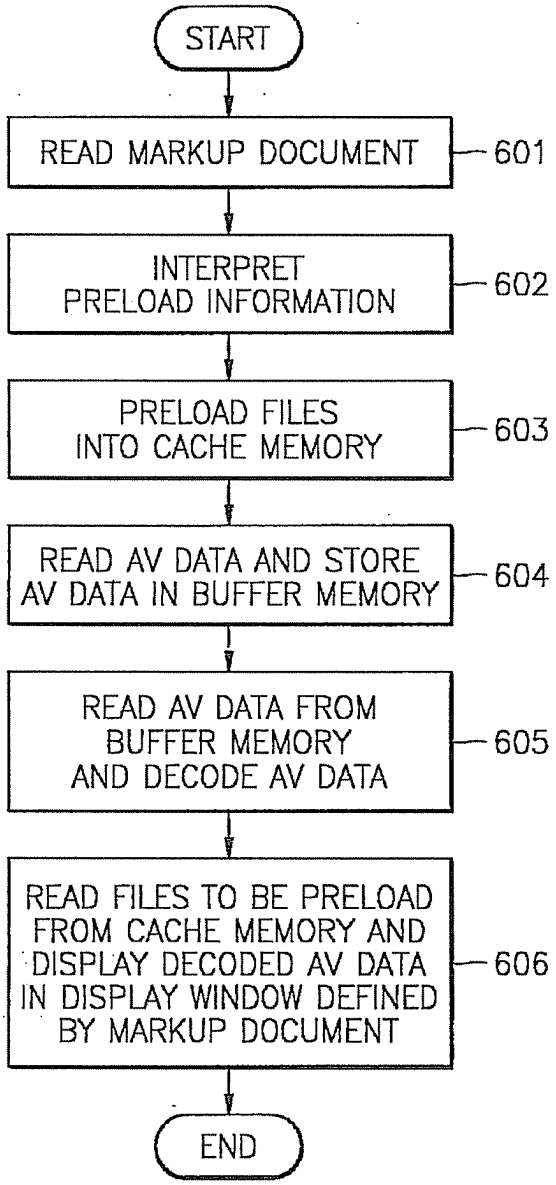

START
READ MARKUP DOCUMENT
601
INTERPRET PRELOAD INFORMATION
602
PRELOAD FILES INTO CACHE MEMORY
603
READ AV DATA AND STORE AV DATA IN BUFFER MEMORY
604
READ AV DATA FROM BUFFER MEMORY AND DECODE AV DATA
605
READ FILES TO BE PRELOAD FROM CACHE MEMORY AND DISPLAY DECODED AV DATA IN DISPLAY WINDOW DEFINED BY MARKUP DOCUMENT
606
END

FIG. 7

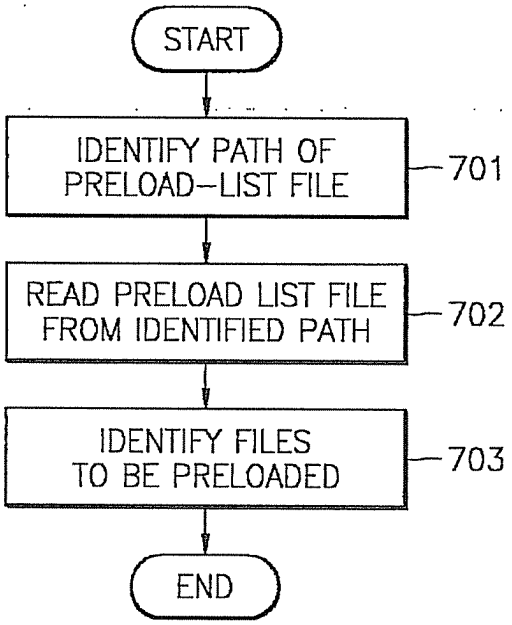

START
IDENTIFY PATH OF PRELOAD-LIST FILE
701
READ PRELOAD LIST FILE FROM IDENTIFIED PATH
702
IDENTIFY FILES TO BE PRELOADED
703
END

FIG. 8

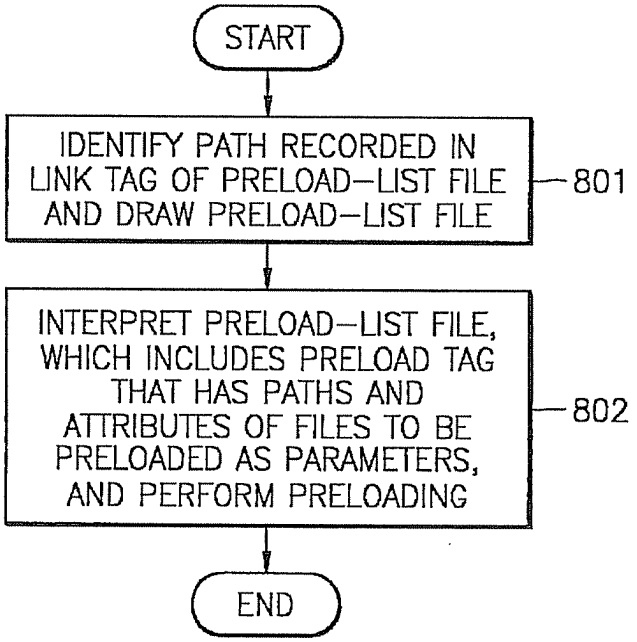

START
IDENTIFY PATH RECORDED IN LINK TAG OF PRELOAD-LIST FILE AND DRAW PRELOAD-LIST FILE
801
INTERPRET PRELOAD-LIST FILE, WHICH INCLUDES PRELOAD TAG THAT HAS PATHS AND ATTRIBUTES OF FILES TO BE PRELOADED AS PARAMETERS, AND PERFORM PRELOADING
802
END

FIG. 9A

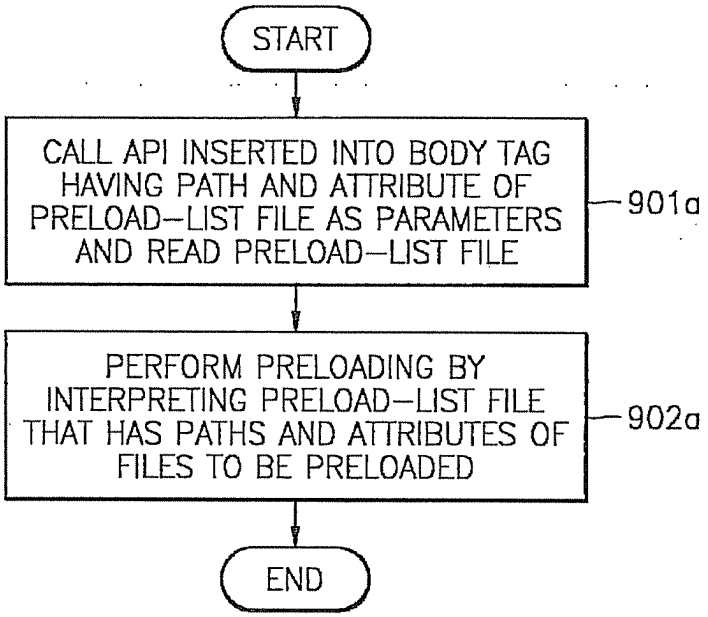

START
CALL API INSERTED INTO BODY TAG HAVING PATH AND ATTRIBUTE OF PRELOAD-LIST FILE AS PARAMETERS AND READ PRELOAD-LIST FILE
901a
PERFORM PRELOADING BY INTERPRETING PRELOAD-LIST FILE THAT HAS PATHS AND ATTRIBUTES OF FILES TO BE PRELOADED
902a
END

FIG. 9B

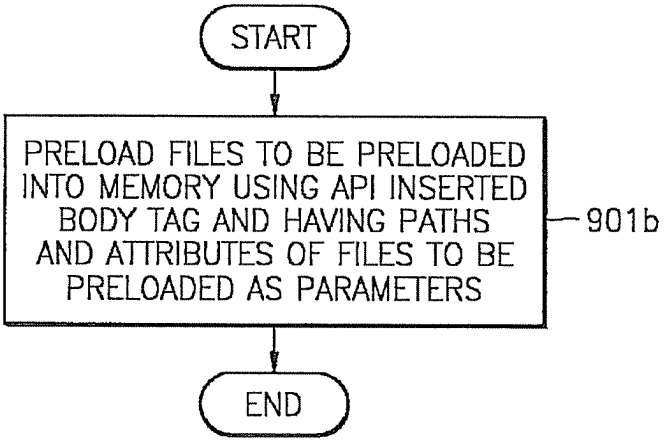

START
PRELOAD FILES TO BE PRELOADED INTO MEMORY USING API INSERTED BODY TAG AND HAVING PATHS AND ATTRIBUTES OF FILES TO BE PRELOADED AS PARAMETERS
901b
END

FIG. 10

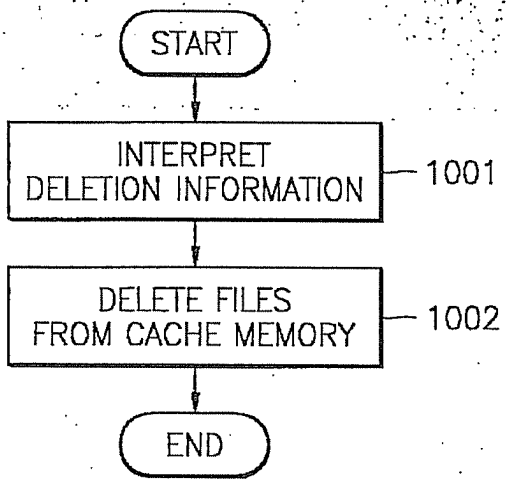

START
INTERPRET DELETION INFORMATION
1001
DELETE FILES FROM CACHE MEMORY
1002
END

FIG. 11

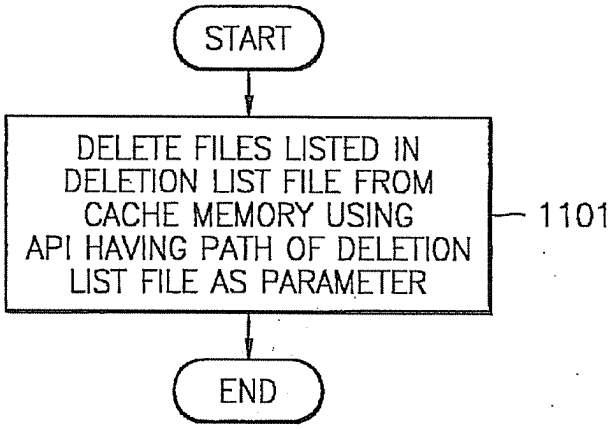

START
DELETE FILES LISTED IN DELETION LIST FILE FROM CACHE MEMORY USING API HAVING PATH OF DELETION LIST FILE AS PARAMETER
1101
END

FIG. 12

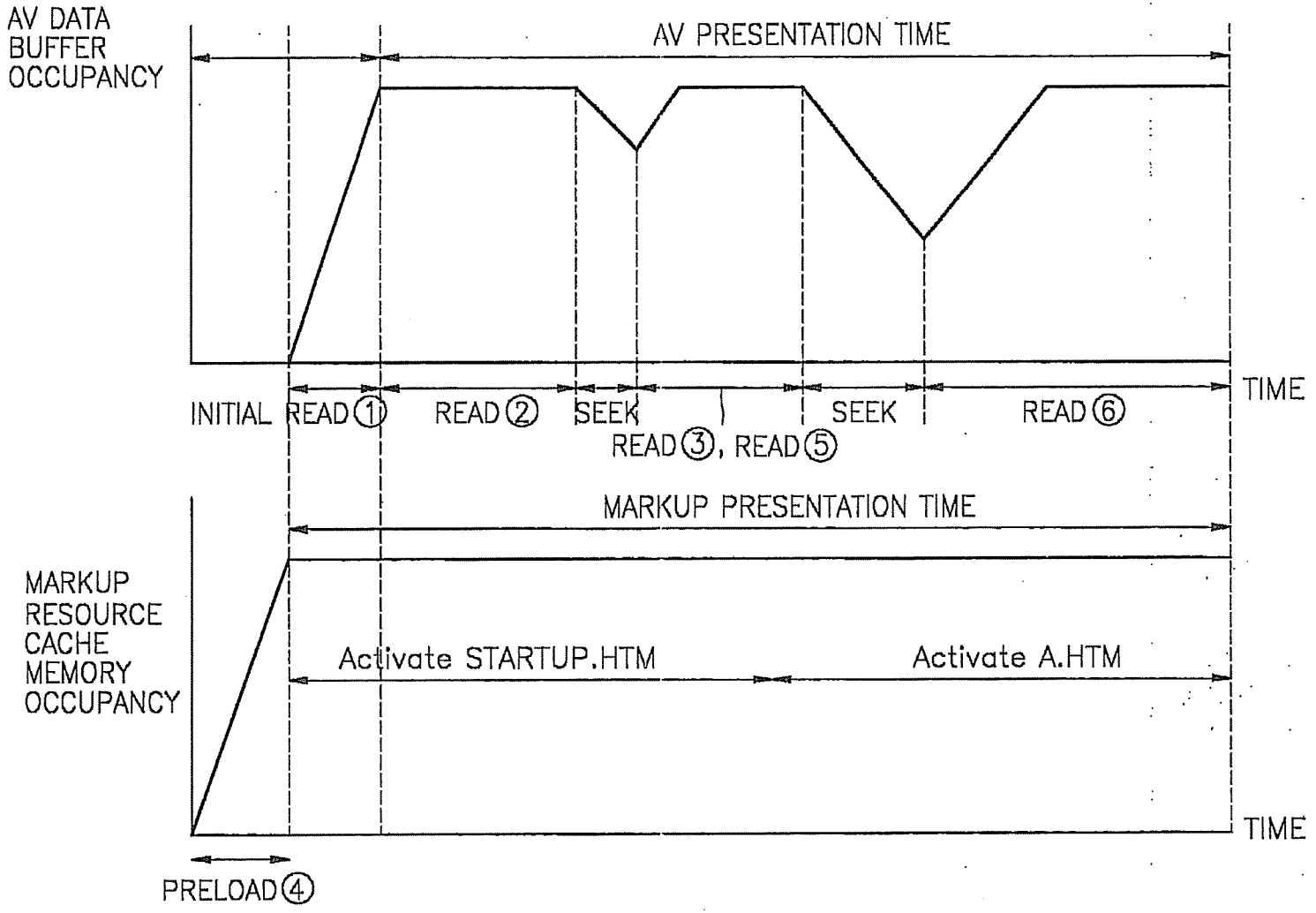

AV DATA BUFFER OCCUPANCY
AV PRESENTATION TIME
TIME
INITIAL READ ①
READ ②
SEEK
READ ③, READ ⑤
SEEK
READ ⑥
MARKUP PRESENTATION TIME
MARKUP RESOURCE CACHE MEMORY OCCUPANCY
Activate STARTUP.HTM
Activate A.HTM
TIME
PRELOAD ④

FIG. 13

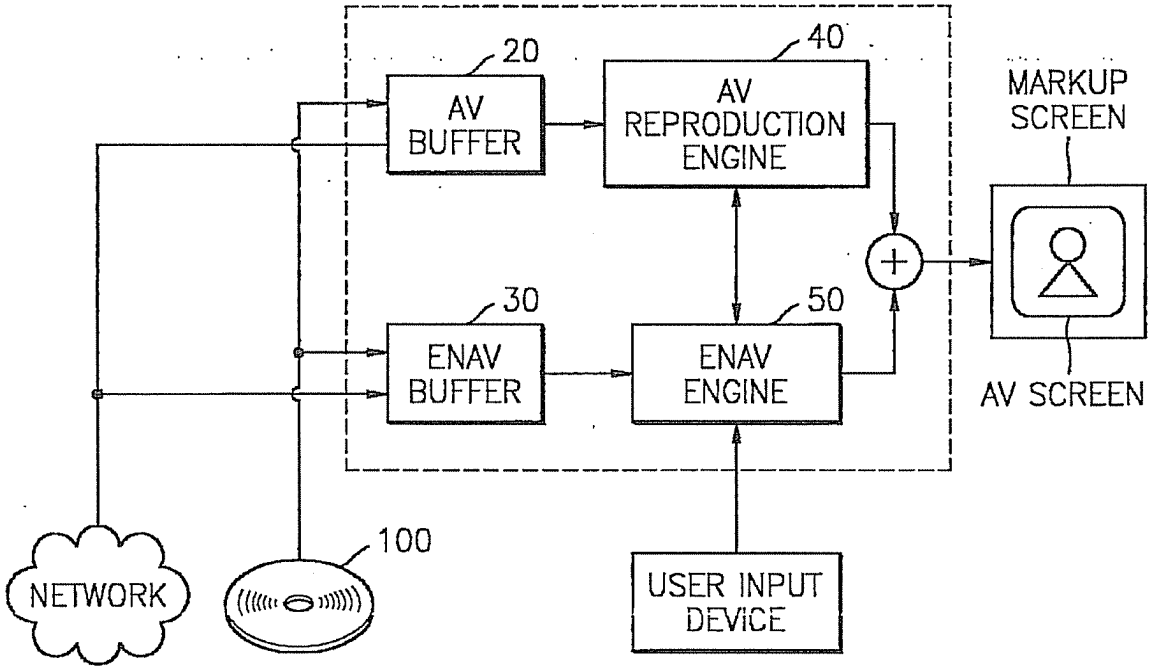

20
AV BUFFER
40
AV REPRODUCTION ENGINE
MARKUP SCREEN
30
ENAV BUFFER
50
ENAV ENGINE
AV SCREEN
NETWORK
100
USER INPUT DEVICE

FIG. 14

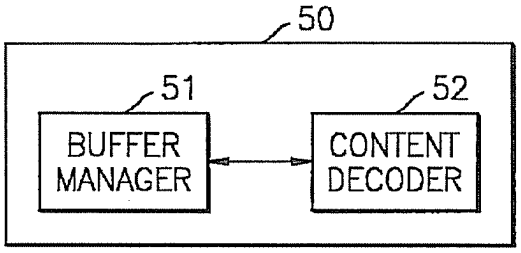

50
51
BUFFER MANAGER
52
CONTENT DECODER

FIG. 15

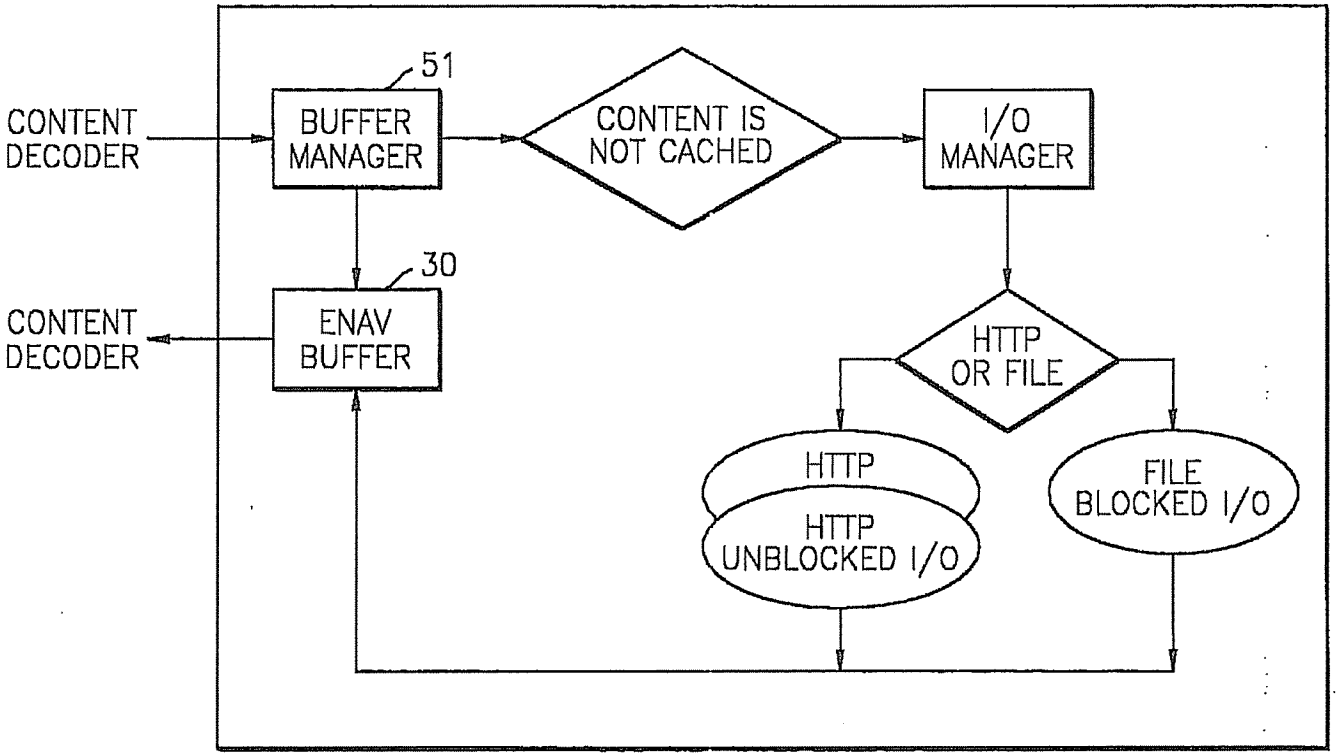

CONTENT DECODER
51
BUFFER MANAGER
CONTENT IS NOT CACHED
I/O MANAGER
30
CONTENT DECODER
ENAV BUFFER
HTTP OR FILE
HTTP
HTTP UNBLOCKED I/O
FILE BLOCKED I/O

FIG. 16

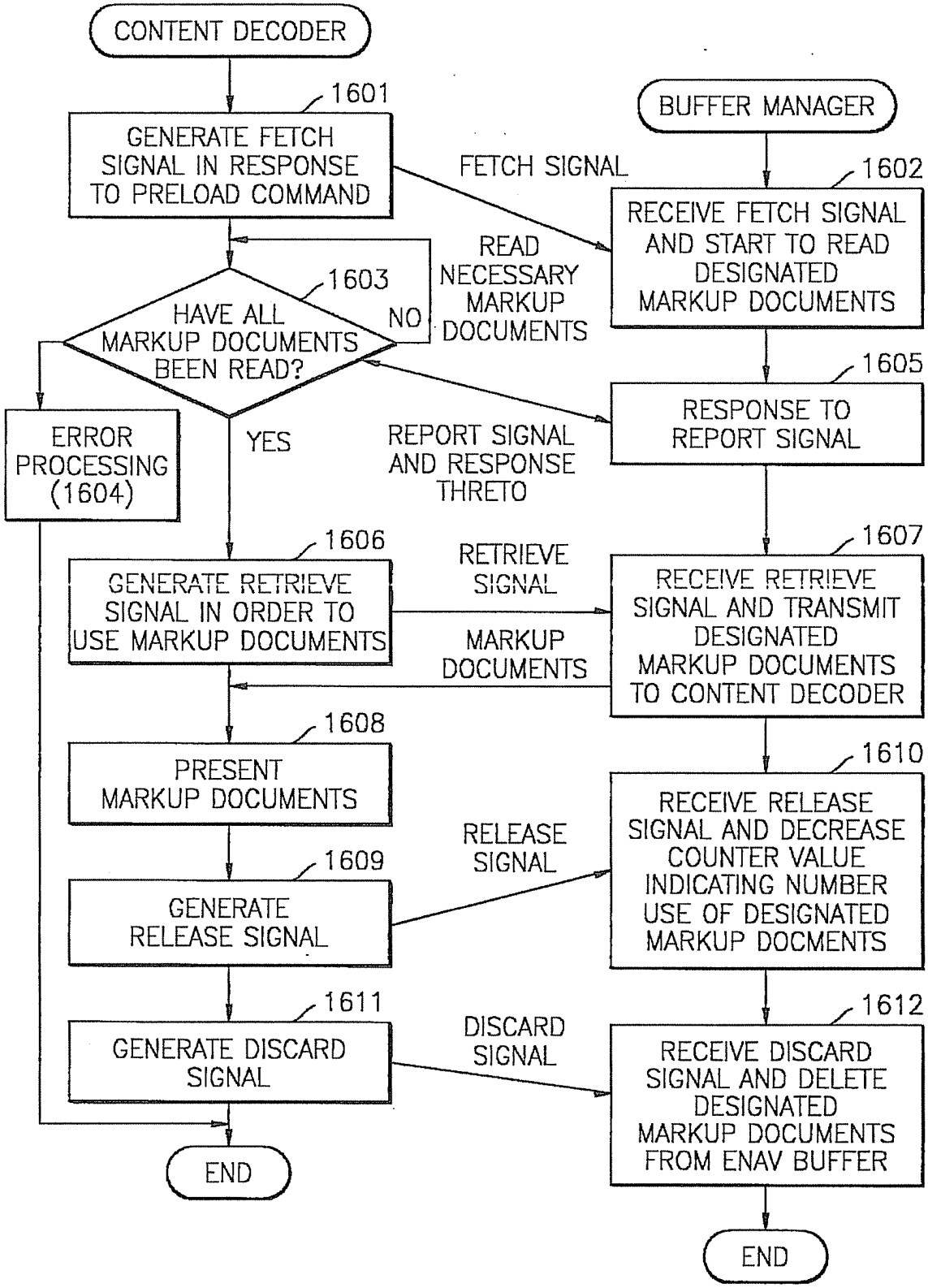

CONTENT DECODER
1601
GENERATE FETCH SIGNAL IN RESPONSE TO PRELOAD COMMAND
FETCH SIGNAL
BUFFER MANAGER
1602
RECEIVE FETCH SIGNAL AND START TO READ DESIGNATED MARKUP DOCUMENTS
READ NECESSARY MARKUP DOCUMENTS
1603
HAVE ALL MARKUP DOCUMENTS BEEN READ?
NO
YES
ERROR PROCESSING (1604)
1605
RESPONSE TO REPORT SIGNAL
REPORT SIGNAL AND RESPONSE THRETO
1606
GENERATE RETRIEVE SIGNAL IN ORDER TO USE MARKUP DOCUMENTS
RETRIEVE SIGNAL
1607
RECEIVE RETRIEVE SIGNAL AND TRANSMIT DESIGNATED MARKUP DOCUMENTS TO CONTENT DECODER
MARKUP DOCUMENTS
1608
PRESENT MARKUP DOCUMENTS
1610
RECEIVE RELEASE SIGNAL AND DECREASE COUNTER VALUE INDICATING NUMBER USE OF DESIGNATED MARKUP DOCMENTS
RELEASE SIGNAL
1609
GENERATE RELEASE SIGNAL
1611
GENERATE DISCARD SIGNAL
DISCARD SIGNAL
1612
RECEIVE DISCARD SIGNAL AND DELETE DESIGNATED MARKUP DOCUMENTS FROM ENAV BUFFER
END
END

FIG. 17

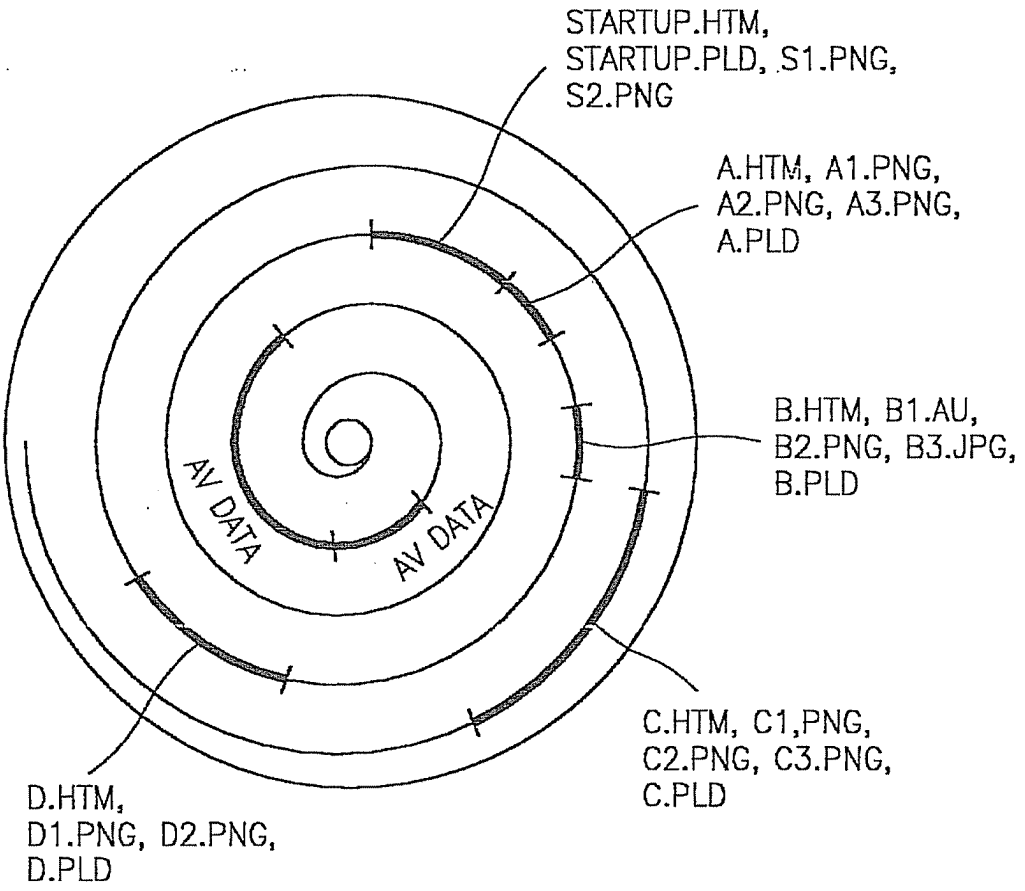

STARTUP.HTM,
STARTUP.PLD, S1.PNG,
S2.PNG
A.HTM, A1.PNG,
A2.PNG, A3.PNG,
A.PLD
B.HTM, B1.AU,
B2.PNG, B3.JPG,
B.PLD
C.HTM, C1,PNG,
C2.PNG, C3.PNG,
C.PLD
D.HTM,
D1.PNG, D2.PNG,
D.PLD
AV DATA
AV DATA

FIG. 18

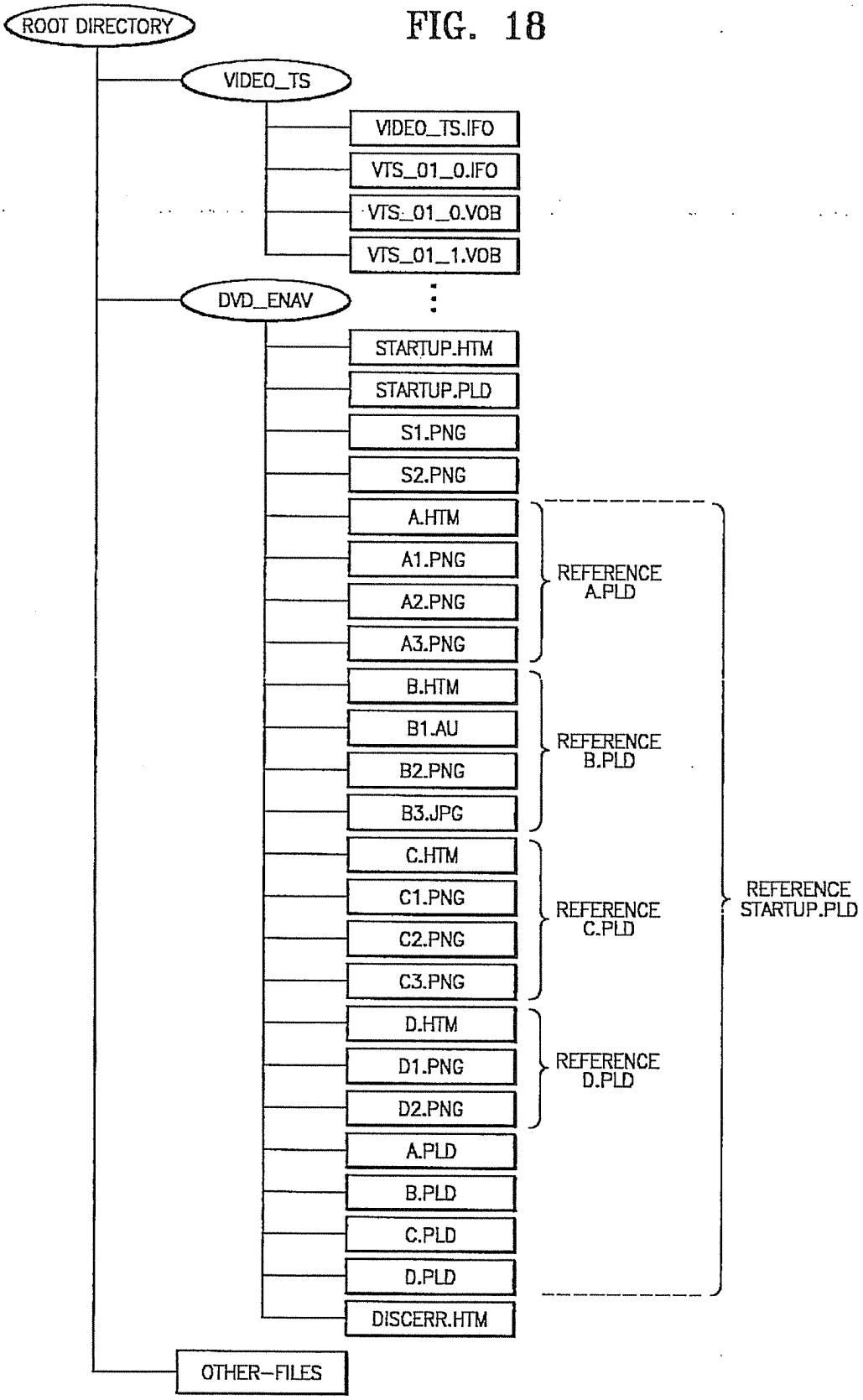

ROOT DIRECTORY
VIDEO_TS
VIDEO_TS.IFO
VTS_01_0.IFO
VTS_01_0.VOB
VTS_01_1.VOB
DVD_ENAV
STARTUP.HTM
STARTUP.PLD
S1.PNG
S2.PNG
A.HTM
A1.PNG
A2.PNG
A3.PNG
REFERENCE A.PLD
B.HTM
B1.AU
B2.PNG
B3.JPG
REFERENCE B.PLD
C.HTM
C1.PNG
C2.PNG
C3.PNG
REFERENCE C.PLD
D.HTM
D1.PNG
D2.PNG
REFERENCE D.PLD
A.PLD
B.PLD
C.PLD
D.PLD
REFERENCE STARTUP.PLD
DISCERR.HTM
OTHER-FILES

FIG. 19

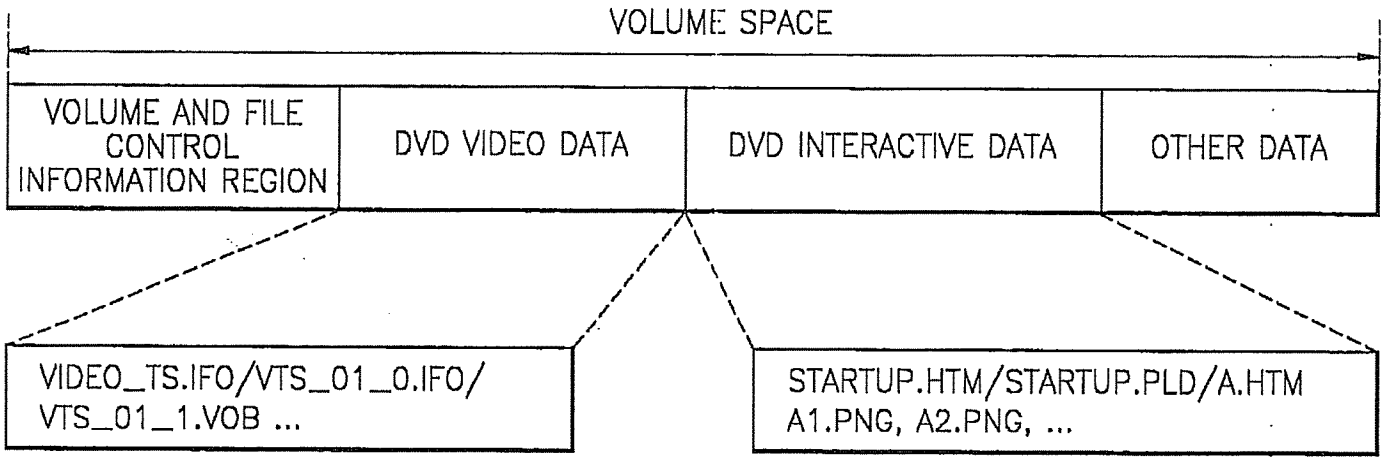
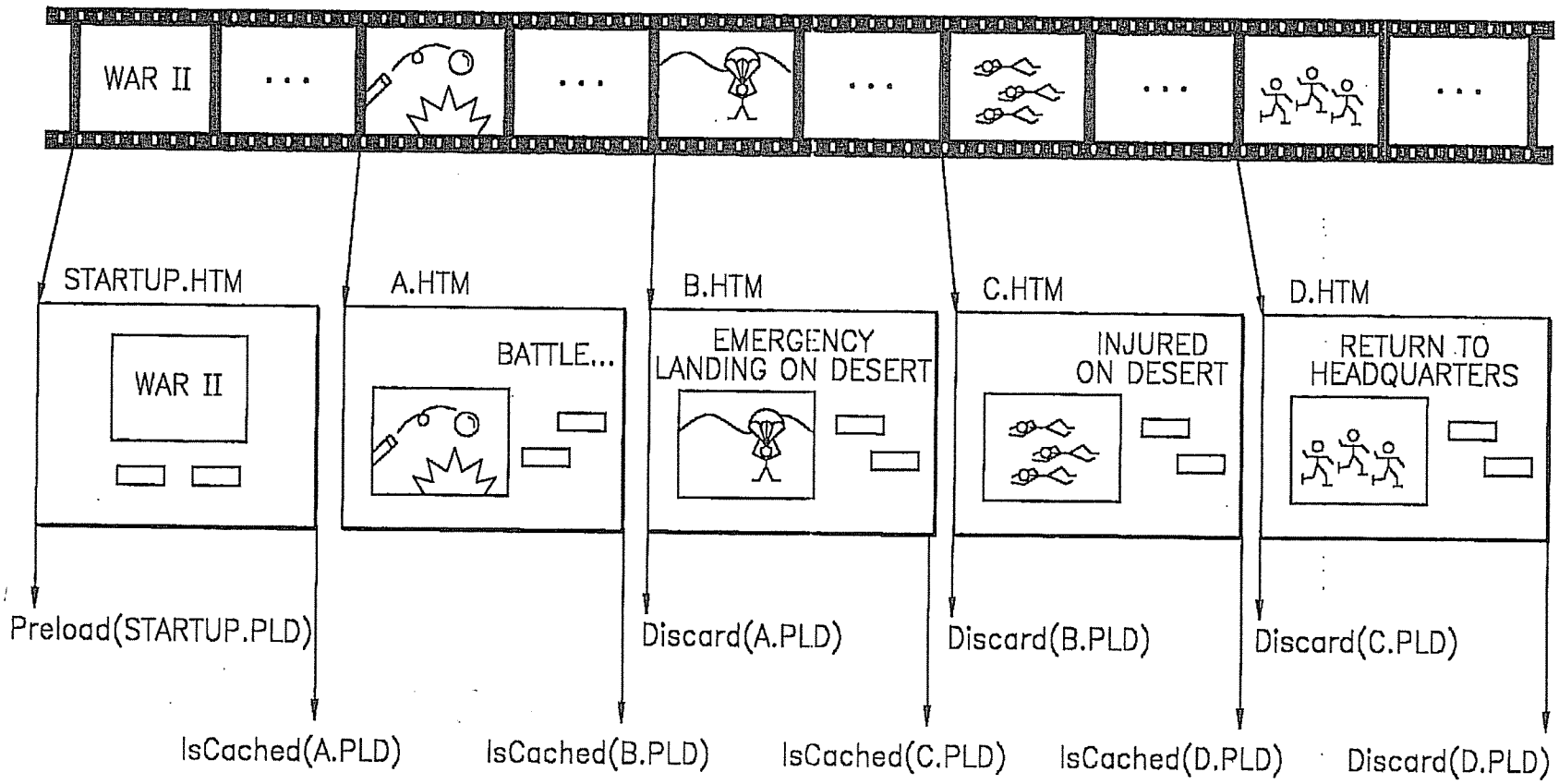

VOLUME SPACE
VOLUME AND FILE CONTROL INFORMATION REGION
DVD VIDEO DATA
DVD INTERACTIVE DATA
OTHER DATA
VIDEO_TS.IFO/VTS_01_0.IFO/
VTS_01_1.VOB ...
STARTUP.HTM/STARTUP.PLD/A.HTM
A1.PNG, A2.PNG, ...

FIG. 20


WAR II
STARTUP.HTM
A.HTM
B.HTM
C.HTM
D.HTM
WAR II
BATTLE...
EMERGENCY LANDING ON DESERT
INJURED ON DESERT
RETURN TO HEADQUARTERS
Preload(STARTUP.PLD)
Discard(A.PLD)
Discard(B.PLD)
Discard(C.PLD)
IsCached(A.PLD)
IsCached(B.PLD)
IsCached(C.PLD)
IsCached(D.PLD)
Discard(D.PLD)